Early validation of satellite COTS-on-board computing systems

Philippe Cuenot, Paul Bouche, Robert de Simone, Julien Deantoni, Amin Oueslati

Early validation of satellite COTS-on-board computing systems

P. Cuenot[1][4], P. Bouche[1], R. de Simone[3], J. Deantoni[2], A. Ouestali[1]
1: IRT Saint-Exupéry, System Engineering department, Toulouse, France
2: Université Cote d'Azur, I3S/INRIA Kairos, Sophia Antipolis, France
3: INRIA Kairos, Sophia Antipolis, France

Abstract: The competitive market of nano and micro satellites opens perspectives for use of COTS (Commercial Off-The-Shelf) electronic components. Current modular electronics design for embedded On-Board Computing systems (OBC) is being challenged by the integration of flexible Systems on Chip (SoC). The deployment of generic avionics and user/payload functionalities on these components is becoming increasingly complex, while Quality of Service must remain compliant with demanding requirements. It is therefore most important to estimate/evaluate those properties as early as possible, regarding a given application's deployment on a given COTS-based architecture. Model Based System Engineering (MBSE), while a leading practice in architecture description, may still require further study on its use for early evaluation, especially regarding analysis of emerging behaviors and qualitative model-based mapping of applicative functions onto architectural platform.

In this paper, we present methods to enhance MBSE design, helping the designer in evaluating candidate mappings and design choices by providing concrete quality measures. We focus on two aspects that were identified as critical in the ATIPPIC IRT project: first, the cost and conflicts in data communications in on-board and peripheral interconnects, which has a bottleneck impact on mapping choices; second, the availability of functions in case of resource failures (from solar radiations), to validate fault-mitigation techniques and estimate the (un)availability of the OBC system.

We illustrate the approach on a simplified satellite model, abstracted from a design conceived in the ATIPPIC IRT project.

Keywords: MBSE, Virtual engineering and Simulation, FPGA, Dependability.

1. Introduction and Motivation

The space industry is opening to the trends of micro- and nano-satellites, meant to cover the market of satellite-based services. Potential embedded applications, from broadband internet to specific earth observations, will require low-cost, flexible and reliable electronics architecture to keep competiveness. Modular and integrated electronic architecture, a paradigm already established in other industries, is now crawling into aerospace industry, in the hope to build affordable products using Commercial Off-The-Shelf electronic components (COTS) for greater integration. In particular, modern Systems on Chip (SoC) involves more diverse computing power (CPU, FPGA…), incurring flexibility of application deployment, and possible On-Board Computing system (OBC) reconfiguration.

Of course, this flexibility comes at the cost of numerous design space alternatives. To fully exploit the benefit of SoC performance, a key issue is to devise efficient component mapping (in physical allocation and temporal scheduling).This includes the impact of OBC and peripheral SoC communications interconnects: communication conflicts (bottleneck interferences) may drastically downgrade time responsiveness of applications. The ambition is to estimate **conflicts (interferences) and congestions** of SoC interconnects and to estimate bottleneck latency effects on function execution. Another important challenge in aerospace domain is the need for **dependability/availability**. This is especially important regarding sensitivity of standard SoCs to cosmic radiations since they are not physically hardened. Therefore fault detection, isolation and recovery (during FDIR) must be introduced to optimize operational (safe) availability, itself a dual requirement (fixing failures must not impair too long operationally). It becomes very important to evaluate impact of safety mechanisms on an architecture including effects on bus congestion. Here we target the estimation of operational availability considering effects of fault mitigation inside and around the SoC.

One of our main objective was to use the same architecturo-functional modeling platform as a common basis to build extensions on Performance for congestion analysis and Safety for availability analysis. Furthermore, we aim to exploit this modular structure to be able to localize precisely the distinct modeling requirements facing distinct domain expertise, which could become rather independent inputs that distinct experts concentrate on without overview on global impact. Modeling consistency between viewpoints is a big challenge to foster potential dialog between local experts under supervision of the general system design architect.

Our paper is structured as follows. Section 2 introduces related methods on Model Based System Engineering and Analysis used for availability and performance early evaluation. Section 3 documents the

[4] Seconded from Continental Automotive France, Toulouse France

motivational setup defined by COTS-based OBC design within the IRT ATIPPIC project. Section 4 introduces the proposed method for consistent analyses of congestion and availability. Section 5 provides insight results respectively for congestion evaluation and availability analysis. Finally, the conclusion in chapter 6 reports feedback on our approach and draws perspective for future work.

2. Related Work

Model Driven Engineering, introduced to assist design in the early stage of development flows, is becoming a common practice in some system engineering disciplines [1]. System engineering later combined MDE with other domain-specific modelling paradigms to reach Model Based System Engineering (MBSE), and support analysis and description of both system architecture and functionality (and their mutual mapping). The OMG standardized the SysML[1] profile, nowadays available in numerous commercial tool, and the Capella[2] open initiative provides the Arcadia/Capella [2] modelling environment. AADL is a third example of such MBSE solutions. From ,a software point of view, they all provide basic means to describe tasks and functions of an application, as well as architectural block diagrams for execution platforms, and simple means to describe candidate mapping allocations between them. They all also require modelling language extensions to provide the extra information and annotations required to cover domain-specific analysis of different objective domains (performance, consumption, heat, memory print, safety, security, and so on). The language extension must be complemented by tool viewpoint driven approach as recommended by ISO/IEC 42010 standard. The UML/SysML modeling language extension led to the OMG UML MARTE[3] profile, which offers specializations for Non-Functional Values, for real time application descriptions, and including hardware component descriptions.

In this work we propose to extend Capella by providing additional viewpoints associated with a formal behavioral semantic to analyze Physical Architectures. The selection of Capella was made as it complies with practices of some ATIPPIC partners. Additionally the MARTE Hardware Resource Management (HRM) concepts was used as primary inspiration for language extensions.

Regarding *interconnect congestions*, other solutions could be considered, at least partly. For example, TTool\Diplodocus [3, 4], is a modelling tool based on UML/SysML developed by Telecom ParisTech. As in other so-called Platform-Based Design (PBD) hardware/software co-design tools, design space exploration may be conducted through model translation and then external simulation based on the SystemC[4] hardware simulation language. Bus congestion analysis may be performed, but requires the existence of predefined SystemC blocks. The same construction applies to the commercial-strength Platform Architect[5] tool from Synopsys. PBD using SystemC generally demands hardware design skills rather than system's one (at the level considered in this work). Finally, there exist network simulation frameworks such as NS-3 [5] and OMNET++ [6] specialized in network protocol accurate simulation. This kind of frameworks does not consider hardware platforms close to our domain with regard to SoC decomposition.

On the *availability* assessment calculation side, a few commercial or in-house solution exists. Based on Model Based Safety Analysis (MBSA), they are capable to assess safety but also availability of critical embedded system. The closer tool from our goals and partnerships are based on the Altarica language, as per see SiMFIANeo [7] (developed by APSYS and integrated in Eclipse like Capella) or the open source initiative Altarica 3.0 [8] developed by IRT SystemX[6]. Both allow a system description using the Altarica syntax, and then performing stochastic simulations to evaluate availability. In practice, the Altarica modelling stays at a functional system level, fault propagation occurs between functions and within hardware components. Actual practice availability analysis are performs on full system level, while our study focus on low level SoC including its local reconfiguration. Evaluation with stochastic simulation may require long computing time often not compatible with interactive design). Moreover, tool development efforts for an integration in a single evaluation environment have to be considered. We concentrated our study on an integrated solution that keeps a unique Capella modelling environment. That way, we directly handle a unique system model using additional and existing relationships and properties.

3. Motivational Use-case Set-up

The satellite electronics use-case architecture consists of 3 main layers: a) the COTS SoC is a Xilinx Zynq7000 processor, with a dual-core CPU and FPGA computing power; b) two such SoCs are integrated in a Hyperion OBC Board, together with rad-hardened controllers and peripherals; c) finally the global architecture provides satellite communication features and sensor/actuator systems (cameras, antennas, engines…). The flexibility in mapping computations to CPU or FPGA poses constraints on the internal communications and data transfers. The two-SoCs redundancy allows various schemes of fault mitigation (which may differ on FPGAs and on CPUs). Reducing interconnect congestion, characterizing the satellite availability under given mitigation schemes

[1] https://www.omg.org/spec/SysML
[2] https://www.polarsys.org/capella/
[3] https://www.omg.org/omgmarte/
[4] http://accellera.org/downloads/standards/systemc
[5] https://www.synopsys.com/verification/virtual-prototyping/platform-architect.html
[6] https://www.openaltarica.fr/

and analyzing influence on each others were the main design requirements that prompted this study.

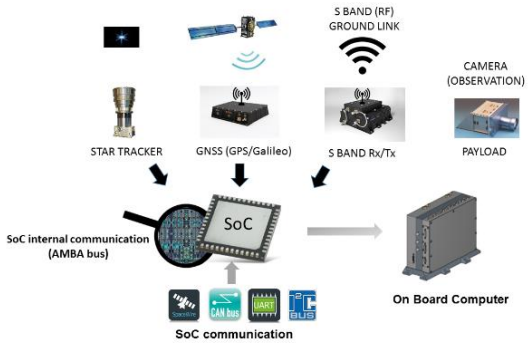

Figure 1: Satellite architecture/OBC overview

On the applicative side, standard avionic functions manage the satellite: *Star Tracker* determines the attitude, *GNSS* acquires the position, *On-Board Time* maintains timing accuracy, *AOCS* controls the attitude and trajectory, and *Telemetry/Telecommand* manages RF communications with the ground. In addition, a Payload application realizes custom user missions (generally, earth observation with image acquisition, compression, storage and ground downloading). These functions are each mapped onto all three architectural layers, with computations and memory storage on the SoC, and communications inside and across levels using relevant interconnects (CAN to interface sensors or payload equipment, SpaceWire to interface sensors and actuators such as payload signal acquisition and propulsion/energy control …).

Regarding fault tolerance, the OBC architecture is based on a cold redundant architecture, each channel being composed of a set of two hierarchical watchdogs. The Zynq is supervised by an external Supervisor Unit (SRU), the SRU by a hardware watchdog. The Zynq embeds fault mitigation components, either by triplication of critical hardware IP features in the Programmable Logic (PL), or by embedding a dedicated IP for detection and/or mitigation of FPGA content such as configuration integrity verification (SEM IP) or supervisor, or via additional software function segregated on each core in the Programmable Software area (PS). Full safety and mitigation concepts are not further detailed in this article only an excerpt is considered.

The Zynq architecture set-up for later assessment covers the need of a spectrometry payload. It requires a large data exchange from spectrometry camera and intensive computation for Fast Fourier Transforms (FFT). The mission control unit controls image acquisition via a SpaceWire link. The acquisition IP stores them in memory. The FFT function operates a calibration and FFT computations on the stored images, which are then compressed and stored them in the NAND flash external memory via an I2C link. Mission control also operates image data transfer from the flash memory to the Radio Frequency (RF) communication unit via the TMI LVDS link, this is to enable ground transfer. The mission's challenge is to im-

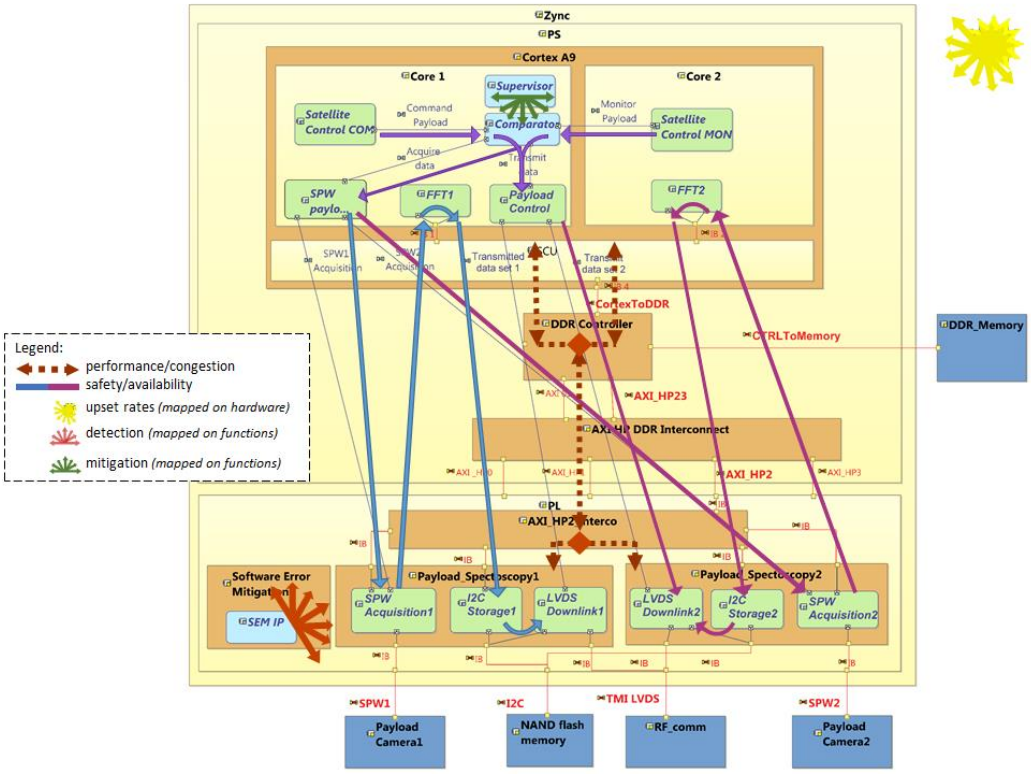

Figure 2: Zynq architecture overview in Capella

prove earth observation performance with the introduction of a second channel of acquisition, computation, compression and transmission with regards to the second payload. The increase in data traffic may affect memory and interconnect components with concurrent access from PL and PS. A bus performance analysis must be completed.

The satellite cruises on a Low Earth Orbit (LEO), it is sensitive to radiations. Indeed, bit flips may occur, caused by Single Event Upsets (SEU) in hardware resources. SEUs can either lead to Single Bit Upsets (SBU) or a Multiple Bit Upsets (MBU). The challenge is then to demonstrate a sufficiently low value of unavailability with this given architecture. For simplicity purposes, the satellite control is abstracted and implemented with a compute (COM) / monitoring (MON) software architecture. Additionally, a SEM IP is introduced for detection of PL configuration integrity stored in RAM (CRAM) with a supervisor function for mission mitigation strategy management. The efficiency of this detection/mitigation set-up in the Zynq architecture must be analyzed to identify its impact on the overall satellite mission's availability and performance.

The corresponding Zynq architecture used for the demonstration and assessment of the proposed methods is depicted in Figure 2. The various modeling concerns regarding availability analysis when mapped on our use case are displayed on Figure 2. The functional chain considered runs through all functional blocks (in green), while extra "safety-functional" blocks are added (in light blue). The main computation splits in two branches (respectively blue and purple arrows), with cascaded functions running either on CPUs (top) or FPGA (bottom). Additional safety features are located on top center (Supervisor and Comparator, and also partly the secondary Satellite Control MONitor), and on lower-left Mitigation block. Every functional block needs proper information on how "raw" radiation upsets locally lead to failure rates in Erroneous/No Data range (recalled with the sun-shaped icon top-right). Quantitative figures about fault detection and mitigation (duration, latency...) will annotate the Supervisor and SEM IP safety functions.

4. MBSE methods for early analysis of SoC deployment

To keep models tractable, while rich enough to support the evaluation of candidate deployments of applications on the SoC with support for domain specific engineering analysis (and further impacts the whole architecture), we need: (i) to rely on an existing (familiar?) modelling framework, (ii) to precisely identify relevant properties to analyze, (iii) to identify (or develop) a tooled environment that may facilitate the analysis and hide most of its complexity to the designer.

For (i) and (iii) justifications we selected Capella Studio, together with the Kitalpha[7] eclipse plug-in, allowing creation of domain-specific viewpoints. Viewpoints may help to reconcile analysis and evaluation of non-functional requirements managed across different disciplines and expertise, while keeping the overall system model consistent. We identified that the physical layer as the adequate modelling layer in Capella to introduce function-to-resource allocations. It corresponds to entities and viewpoints of the Physical Architecture Blank diagram (PAB). At this level, we specialized data models with adequate properties and new viewpoints.

We consider two kinds of evaluation: (i) *static analysis* methods, assuming worst-case conditions and producing fast, coarse results; (ii) *operational analysis*. Relying on dynamic simulation scenario (that may be exhaustive) for more accurate (but computationally costly) results.

To build the correct emerging behavior in (ii), a behavioral semantic must be defined at the appropriate level of abstraction. For this we use the Gemoc[8] studio, also an eclipse plugin (as Kitalpha), especially meant to define formal operational semantic for simulation. Gemoc provides an interpreter, a debugger and a compiler. Its operational analysis semantic complies with data model extensions, allowing property-based calculations (with Java code).

To complete system design analysis we applied the evaluation approaches on our two target fields: (i) **congestion analysis,** mostly here by computing interference and (over)load among the SoC buses, usually built from hardware and software expertise, (ii) **availability analysis**, establishing mission unavailability due to component sensitivity to radiation failures and mitigation mechanisms of the design, involving safety and reliability expertise. The proposed methods are detailed in turn.

Congestion Analysis

The objective of congestion analysis is to be able to predict and bound memory traffic on SoC internal buses and interconnects by identifying their latencies and throughputs. This allows system designers to validate the application function allocation on hardware resources (PS or PL) by estimating the data transfer impact on micro-architectural buses for coverage of application function performances (communication flow compliance and timing latency).

The congestion analysis information was already published in [9]. Below we only summarize principle to understand the evaluation. We also document improvements made to support more complex scenario during the operational method.

We identified which model elements truly influence the interconnect traffic (computations, communications, memory storage). We annotated these ele-

[7] https://polarsys.org/kitalpha/

[8] http://gemoc.org/studio

ments with properties providing dimensions and behaviors of the bus transaction communication. An example is displayed in Figure 3 (with additional methods/functions used by Gemoc for simulation).

The analytic method is a simple and fast method to compute bound values. So in memory transaction, we consider only static configuration, meaning 1) every transactions are atomic, 2) interconnect arbitration is not considered 3) every transactions crossing a bus port interface are concurrent.
For each component execution (task) of the system we computed a *maxDelayTime* corresponding to the maximum time during which a task can be blocked by other tasks using the same interconnect bus port in a period, the hyper period of the system. For each bus we computed *maxInterference* as the maximum time during which the bus can have data blocked on its interface computed in a hyper period, and *load* as occupation of the bus computed in a hyper period. These blocking situations consider worst case conditions on interconnect port due to 2) and 3), so computed value are pessimistic and deserve as first idea for performance evaluation on bus congestion. Kitalpha is used to develop Capella view point including Java code for calculation of these estimations.

The operational method allows to refine previous results based on simulation scenario. We used the Gemoc studio [10] facilities to define an operational semantic for the execution of the Capella PAB diagram extended viewpoint. Initiators of memory transaction are tasks (component execution) with behavior properties defined from data model of Figure 3. We consider a Read, Execute Write task execution paradigm with randomized time execution between its best and worst case value to implement execution variation. Read or Write memory transaction can be blocked by another on-going transaction, delaying task execution. In [9] we considered only atomic memory transactions, the base time for encoding Gemoc simulation step. We now allow to decompose large memory transactions in beats corresponding to burst transactions. Moreover, we extend the scheduling semantic with Least Recently Granted (LRG) and Priority based schemes to support advanced functionalities of interconnect and memory controller components. This evolution covers the abstract representation of QoS services of Zynq interconnect arbitration. Priority settings. It also allows to perform arbitration during Read/Write memory transactions on each burst frame, to respect priority setting in function communication scheduling when large transaction frames are managed.

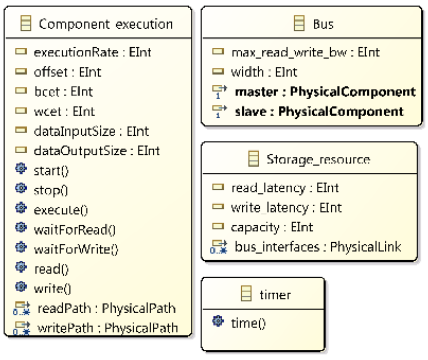

Figure 3: Congestion Domain model

The link between the Capella meta-model extension and Gemoc execution is built with function defined in Figure 3 implementing system behavior and with additional action required to compute bus load and interference. This is a so-called Domain Specific Action (DSA) which is triggered by a Domain Specific Event (DSE). As several occurrences of objects may execute in parallel, Gemoc generates a Model Specific Event (MSE) as an ordered set of event occurrences required to trigger each DSE event instances. Execution order is defined by specifying invariants in CCSL (Clock Constraint Specification Language) and in MoCCML (Model of Concurrency Modeling Language) [11]. MoCCML semantic is implemented by an automata, where each transition is triggered by a time event [12]. The physical time is built according time event, with resolution matching the smallest encoded memory transaction.

According to Zynq QoS specifications, its implementation requires two level of arbitration, first a priority based on master port and then a LRG based for multiple request with same the QoS value. The QoS value is simply abstracted by the transaction decomposition in beats with a fixed burst size. First we added the required properties, respectively priority level and burst length, on the master port of interconnect components using Kitalpha in the viewpoint extension. Then for the priority based policy, arbitration constraints were added in the DSE using OCL invariants at the meta-model level as a part of the semantic. By navigating in the model using OCL queries, we are able to compare the QoS signals of the concurrent communications and grant the bus access to the transaction with the highest priority. Applying this to a concrete model, when loading the model, Gemoc will generate a “prioritymodel” file in which priorities between transactions are specified explicitly. The file will then be given as an input for the simulation engine. During the simulation, when two (or more) concurrent events corresponding to concurrent transactions appears, the choice of the simulation engine will be depend on the priority file. For the LRG arbitration scheme, we used a different approach since it was not meant to be encoded in the semantics but rather in a dedicated simulation policy. In the LRG scheme, priorities are not specified explicitly in the model as their values change over time. Indeed the dynamicity of priorities makes it impossible to give a priority order between the different transactions. Therefore, instead of adding the arbitration to the model’s semantics, we extended the Gemoc concurrent engine with a simulation policy based on the LRG scheme. The algo-

rithm manages choices for multiple requests on arbitration points for different communications on the same bus coming from different paths allowed on the state space of the simulation. It selects the highest priority among the bus requests.
Gemoc is so integrated with Capella Studio to allow simulation of PAB extended viewpoint and display estimation of bus interference and load.

Availability Analysis

In this work, we want to assess the unavailability of each functional chain composing the system and sum the obtained values to get the overall unavailability. The unavailability of a functional chain occurs when it is no longer capable of doing its intended purpose correctly. This happens when one or several critical functions do not behave as intended anymore. In our work, failures are caused by SEUs in hardware component and are reflected through the data exchanged between functional components. These failure can be spread between functions since every data is consumed by a function to output new data and a failure in an input data can lead to a failure in the output data. Ultimately, we want to assess the duration taken by the system to detect and mitigate such failures.

The focus of this analysis lies in evaluating the consequences in terms of functional failures of the aforementioned radiation upsets occurring in hardware. Indeed, we assume that each functional component relies on one or several hardware components to achieve its intended purpose. Therefore, the occurrence of a SBU or MBU in a hardware component, may affect the functional behavior of any logical component relying on it. These failures are potentially propagated from a function to another within the system and ultimately lead to its unavailability. In order to assess this impact, we introduce three abstract data-oriented failure modes for functional components:

- Erroneous Data In Range (EIR) is the result of either the alteration of required data, or instructions leading to a modification of the output data while staying in their functional range.
- Erroneous Data Out of Range (EOR) is the result of either the alteration of required data, or instructions leading to a modification of the output data lying outside of their functional
- Finally, No Data (ND)) is the result of either the alteration of required data, or instructions leading to the logical function to cease providing data

The method described hereafter concerns the analytical analysis. Therefore, some assumptions are required. According to experts, the SEU probability's order of magnitude and the usual failure mitigation duration, we assume that situations where a SEU occurs while a first one has not been detected/mitigated with are very unlikely and are not considered in this paper. Moreover, failures can be volatile, in the way that data are cyclically overwritten and thus any SEU modifying such a data will only last temporarily. Conversely, some SEU will never disappear unless a corrective action is undertaken.

We consider the relation between upsets and functional failures as being purely probabilistic. For a given logical component L, S is the set of sensitive physical components relied upon by L, the probability of a given failure to happen in L is computed as per:

$$p_L(failure) = \sum_{i \in S} \sum_{j \in upset} p_i(j) \times p_i(failure \mid j) \times usageRate(L, i)$$

where, $upset = \{SEU, MBU\}$

$p_i(j)$ is the probability of j happening in i

$p_i(failure \mid j)$ is the probability of $failure$ to happen in i knowing j

$usageRate(L, i)$ is the usage rate of i made by L

The usage rate of a hardware component by a logical component depends on the former's nature. To tackle this we propose, as per the congestion analysis, three hardware resources specializations: communication resources (i.e. bus and I/O controllers), execution resources (i.e. CPUs) and storage resources (i.e. data and instruction memories).

Moreover, failure states can spread across functional components within a functional chain and ultimately lead to its unavailability. Indeed, as mentioned before, some functions within a functional chain are critical to ensure proper functioning. To reflect it in our model, the designer must indicate at least one function deemed critical to assess the functional chain's unavailability. This is happens when due to failure propagation, at least one of these functions has any of its input in any failure mode. We introduce failure propagation truth tables, to reflect this propagation within functions and functional data exchanges. For a component with m input and n output, the truth table expresses the failure state of each of the n output depending on the possible combinations of the m input failure states. Regarding functional data exchanges, the goal is to express the impact of the lower level communication medium's nature on failure propagation. For instance, in case the data propagation uses a shared memory, when no data is output by the sending function, the receiving function will read the previously stored value which, in our model, is an Erroneous Data In Range. Since this case is relatively specific to some means of communication, we propose to apply the identity transformation unless a truth table is provided.

We need to express how these failures are detected and mitigated within the system. The European Cooperation for Space Standardization (ECSS) publishes guidelines regarding radiation effects mitigation [13]. Although, this work utilizes only a subset of the mechanisms described in this document, we aim at providing the capability to model them all. Detection mechanisms either ensure SEU detection directly in

the hardware or their consequences as logical level failures. They are characterized by a list of monitored components, the type of event (a SEU or a failure) detectable, and the detection duration. When a detection happens an event detection signal is sent to a mitigation component. The latter is responsible to trigger recovery actions. We suggest the following syntax to express recovery actions' definition: "*N* occurrences of *detectionSignal* implies *componentList* unavailable for *mitigationDuration*."

N indicates the number of successive *detectionSignal* required before triggering the action, *detectionSignal* is the signal send by a detection mechanism, *componentList* indicates the components affected by the recovery action and the recovery action lasts *mitigationDuration* time. Physical components, logical components and functional chains are the type of component which can be affected by a mitigation action. This mean these components are reinitialized to a nominal state. During the mitigation duration any logical component either directly mentioned or taking part of a listed functional chain or relying on a listed physical component outputs No Data. Therefore, the functional chain might be unavailable during the recovery action depending on the failure propagation.

A function's unavailability contribution is computed using the dedicated algorithm for the analytical analysis, where time is only seen as a parametric value. Indeed, only temporal characteristics are used in this analysis. The main idea is for of each dependency's upset happening to check whether a detection and a mitigation mechanism exist. If it does to assess, based on the impacted components, if it corrects the upset and whether the action generated unavailability. Then, the upset is converted into the various failure mode for the function under scrutiny. The capability to detect and mitigate each failure state is assessed for each functional component from the function under scrutiny and going downstream the functional chain. The mitigation duration taken is the max duration found. The summation of this computation for every functional chain provides the system's unavailability.

Relevant information for this analysis originates from several --quite different-- sources of engineering expertise. In our approach we wish to localize the distinct contributions and promote their modular design. We propose a viewpoint-oriented model where each of the contributors input the assessment's required information independently. Radiation specialists should provide sensitivity rates towards SBUs and MBUs; while hardware engineers provide the bandwidth (communication resources only) and size (storage resource only); software engineers consider the dependencies of functional components towards hardware components along with their usage (throughput for a communication resource, usage percentage for an execution resource or memory size for a storage resource) and radiation upset to failure conversions; safety engineers should provide the detection and mitigation mechanisms.

5. Assessment results

Congestion assessment

The scenario for the use case assessment, presented in section 3, aims to demonstrate that the OBC architecture is able to support the increase of data transfer in SoC internal buses due to the introduction a second spectrometry channel and to co-exist with detection/mitigation safety functions. We concentrated our study on buses accessing the *DDR memory Controller* because of the large size of added data transfer (150 Mb every second). The memory controller (and DDR memory) is a shared resource between logical functions, for data transfer. Depending of function scheduling points (offset parameter value for execution component), bus load and latency may be strongly affected as we will demonstrate in this example. Every memory transaction used to configure IPs in the PL area are ignored, since they correspond to a low volume of data (only some sporadic frames). Payload IPs are connected to a dedicated interface port of the DDR interconnect, different from the one of the satellite control IPs. The purpose is to ensure safety via segregation of memory transaction flows. Traffic on these bus interfaces is then ignored. So we analyze congestion and latency on *CortexToDDR, AXI_HP2 and AXI_HP23* bus of Figure 2 and impact on delayed function execution.

The first evaluation consists in analyzing dual spectrometric mission, build with respective logical blocks *Satellite Control COM* (rate 10Hz, offset 100ms, operating control with R/W of 300Kb of data), *SW payload driver* (rate 1Hz, offset 0, transmitting SPW control with R/W of 1Kb of data), *SPW Acquisition1&2* (rate 1Hz, offset1 0ms & offset2 200ms, acquiring data from SPW and writing in DDR 150Mb of data), *FFT1&2* (rate 1Hz, offset1 200 & offset2 300, reading 150Mb of data, performing FFT and compressing them to 3Mb and writing data in DDR), *I2C Storage1&2* (rate 1Hz, offset1 900ms & 1100ms, reading 3Mb of data from DDR and writing them in NAND flash). LVDS Downlink1&2 are not considered in this analysis as manipulated data are directly copied from the NAND flash to the RF communication component by point to point buffered data communication, and so are not sharing DDR component. The minimum data size configured in the tool is 1 Mb (actual tool constraints) and memory transactions are managed in 4Kb burst size (maximum size of Zynq AXI specification). Memory controller is configured with LGR policy on master port as default configuration, so call fair configuration. In a second step, we configured high priority for memory transaction of safety related function (Supervisor, Comparator and Satellite Control MON&COM) to mimic priority control on communication scheduling. All bus are configured in 64bits width

and 1.2 Gb as maximum bandwidth. We assumed that memory controller is “enough” performant to guarantee full bus bandwidth on CTRLToMemory bus for access to DDR memory.

The analytic evaluation computes the bus load value and the maximum interference rate for each bus, as well as the maximum delay time and communication transfer time for each function involved in the analysis. The results are directly displayed in the Capella viewpoint (see Figure 4). Values provided are worst case values, since dynamic conditions such as function scheduling offsets or communication transaction priorities are not considered.

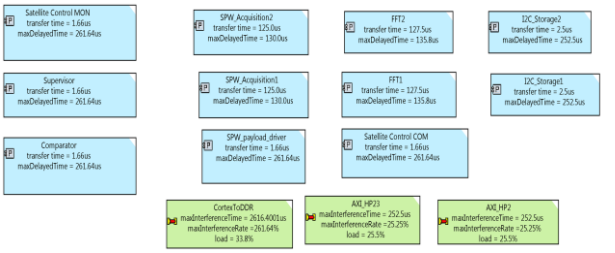

Figure 4: congestion analytics results display

When introducing safety blocks, *Satellite Control MON* (rate 10Hz, offset 100ms, operating control monitoring with R/W of 300Kb of data), *Comparator* (rate 10Hz, offset 150ms, mitigating COM/MON redundancy with R/W of 300Kb of data) and Supervisor (rate 10Hz, offset 200ms, mitigating hardware failure with 20Kb of data) the bus load and inferences increase on CortexToDDR bus, while remaining constant on other AXI_HPx buses (see results in Table 1). Indeed, the safety components are only introduced on the PS side, connected to memory via the CortexToDDR bus, and additional data transfers (6Mb) are limited compare to initial transactions (310Mb). We can first conclude that the proposed architecture is capable of handling the data bandwidth with a dual spectrometry mission including satellite control integrity services. Function delays need to be bounded with more precise values (and especially safety ones).

	CortexToDDR	AXI_HP23	AXI_HP2
Dual spectrometry payload (Analytic)			
MaxInterenceRate	25.6%	25.25%	25.25%
Load	28.8%	25.5%	25.5%
Dual spectrometry payload with safety function (analytic)			
MaxInterenceRate	26.16%	25.25%	25.25%
Load	33.8%	25.5%	25.5%
Dual spectrometry payload (Operational and Fair)			

Table 1: Congestion analytics results

The operational analysis method brings more precise results as we can see on Figure 5, with results variation over time. To minimize bus congestion, we performed an exploration of scheduling offsets. This exploration was performed thanks a DSL capable to define simulation scenario (see **[9]** for DSL information). The explored configurations are performed by shifting the offset of the main consumer in data transactions on the second payload, such as Acquisition2 and FFT2 functions. Offset varies respectively between [0-100] and [200-360] with an exploration step of 20 for each. Every simulation results are accessible at the following URL:

https://github.com/jdeantoni/ERTS2020_artifacts.

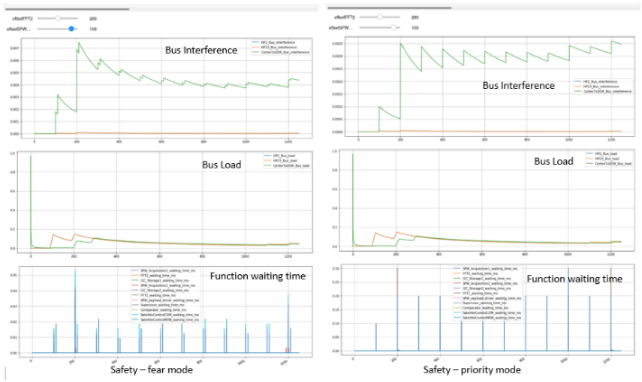

Figure 5: Congestion operational results

The analysis was performed in fair transaction mode (LRG) left side of the Figure 5 (payload with safety) and with high priority setting for safety functions on the right side. We record 4% of interference on CortexToDDR (green curve) and maximum 10% of bandwidth after stabilization (green curve). Produced results are clearly more precise than analytical methods and allow to interpret the waiting time for each function. The impact of priority setting shows that bus interference is increased, since safety functions delay large frames, getting more impact on the overall interference. We demonstrate that such operational analysis allow to better analyses provisional bus congestions thanks to a scheduling proposed design on the system architecture.

Availability assessment

This section aims to demonstrate, using the analytical method defined in section 4, on the use case defined in section 3, how the introduction of detection and mitigation mechanisms reduces the system’s unavailability, in the use case under study the Zynq is the system. In this assessment, we only study sensitivity towards SBUs, for simplicity purposes. The following data are used regarding sensitive hardware components, the sensitivity is expressed in number of SBU happening per device per day: CPU Cores 3.52E-3 SBU/day, DDR 2.25E-2 SBU/day Memory, FPGA CRAM 4.57E-2 SBU/day and NAND Flash 3.06E-3 SBU/day. In terms of functional to hardware component dependency, the following is used: Satellite Control COM depends on Core 2 (15%) and DDR Memory (996 kB), Satellite Control MON depends on Core 1 (15%) and DDR Memory (996 kB), SPW Payload Drive depends on Core 1 (5%) and DDR Memory (100 kB), SPW Acquisition 1 and 2 depends on CRAM (20 Kb) and DDR Memory (80 kB), FFT1 depends on Core 1 (20%) and DDR Memory (150 Mb), FFT2 depends on Core 2 (20%) and DDR Memory (150 Mb), I2C Storage 1 and 2 depends on CRAM (10 kB) and

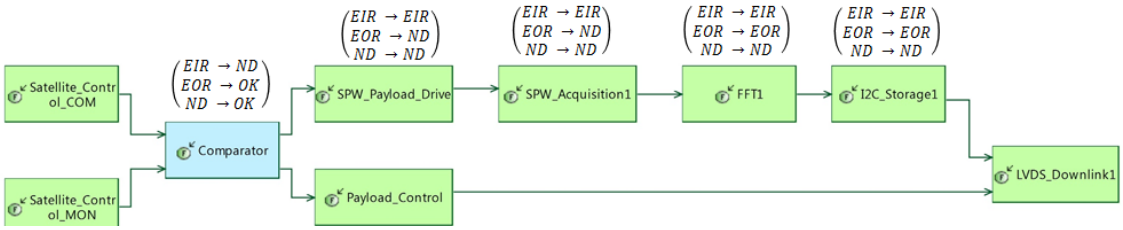

Figure 6: Failure Propagation Description

DDR Memory (3 Mb), Payload Control depends on Core 1 (10%) and DDR Memory (50 KB), LVDS Downlink 1 and 2 depends on CRAM (10 kB), DDR Memory (80 KB) and NAND (3Mb), the Comparator depends on Core 1 (5%). The upset to failure rates are taken into account but are not shown in this due to a lack of space. Just as an example, for Satellite Control COM and MON we have SBU in Core leading to 80% of Instruction Errors with 100% of ND, to 20% Data Error which volatile (because happening in registers which are overwritten frequently) with 50% of EIR, 50% of EOR and SBU in DDR leading to 100% Data Errors with 80% of EIR, 20% of EOR.

In terms of failure propagations, every data exchange between the PS and PL uses a shared memory. Therefore data exchanges between SPW Payload Drive and SPW Acquisition (1 or 2), SPW Acquisition 1 (resp. 2) and FFT1 (resp. 2), FFT1 (resp. 2) and LVDS Downlink 1 (resp. 2), the failure mode propagation transformation converts No Data in Erroneous Data In Range. The failure propagation in logical functions is depicted in Figure 6. It represents only one image acquisition channel, nevertheless the same propagation occurs in the other channel. In this figure, OK stands for No Failure. Due to the single SEU occurrence assumption, in this use case, the Comparator always only has one input data in a failure mode, the other always being OK. Moreover, it outputs the same data on both ports. Hence, the simplified table shown in the figure.

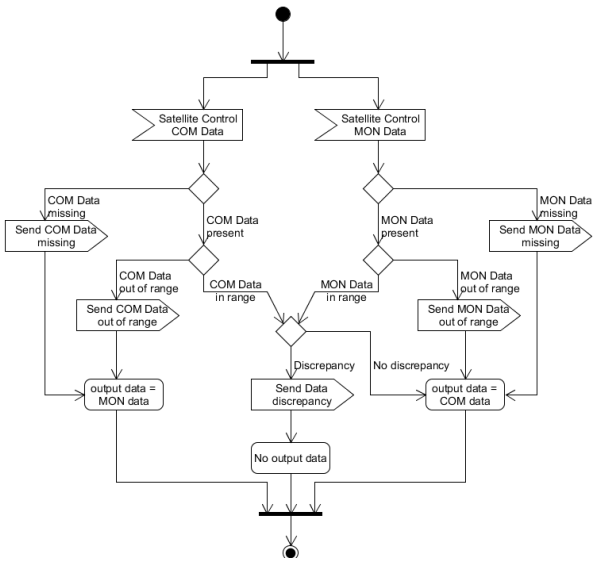

Figure 7: Comparator description

Several safety mechanisms are introduced, they are described hereafter. An internal SBU protection mechanism is deployed in the NAND Flash (data triplication) which means it is not considered sensitive with regard to unavailability computations. The comparator's behavior is depicted in Figure 7. Since it outputs a correct data when one of the input data is either EOR or ND, it acts as a failure isolation mechanism. It also sends failure detection signals to the Supervisor and hence is a detection mechanism. The SEM IP monitors the FPGA's CRAM for SBU detection. The maximum duration required to detect an upset in the CRAM is 6.86s.

The supervisor also has a dual role in the system. It acts as a watchdog towards functions within the CPU, therefore being both a detection mechanism for No Data output by these functions and a mitigation mechanism. It is therefore the only mitigation mechanism in this use case and thus will receive every detection signal. Its recovery actions are the described hereafter:

2 times SEU Detected implies FPGA unavailable for 8s
2 times COM (or MON) Data Missing implies CPU unavailable for 3s
2 times COM (resp. MON) Data Out of Range implies Satellite Control COM (resp. MON) unavailable for 3s
3 times Data Discrepancy implies Satellite Control COM and MON unavailable for 3s.
1 time Comparator or SPW Payload Drive or FFT1 or FFT2 or Payload Control No Data implies CPU unavailable for 10s.

A Capella viewpoint has been developed to input the required data mentioned beforehand. Using the availability analysis method described before on the system with and without safety mechanisms, the following results are computed assuming a 1 year mission.

	Unavailability without safety mechanisms	Unavailability with safety mechanisms
Satellite Control COM	35.28 days	1.88E-5 days
Satellite Control MON	N/A	1.88E-5 days
Comparator	N/A	5.25E-6 days
SPW Payload Drive	117.83 days	5.25E-5 days
SPW Acquisition 1	137.8 days	2.08E-4 days
SPW Acquisition 2	137.8 days	2.08E-4 days
FFT1	47.29 days	2.23E-5 days
FFT2	47.29 days	2.23E-5 days
I2C Storage 1	68.73 days	1.04E-4 days
I2C Storage 1	68.73 days	1.04E-4 days

Payload Control	23.53 days	1.05E-5 days
LVDS Downlink 1	68.8 days	1.04E-4 days
LVDS Downlink 2	68.8 days	1.04E-4 days

The results of the computation without safety mechanisms clearly highlights that local contributions can only be summed to compute the overall unavailability when they occur independently since the summation is greater than 365 days.

When safety mechanisms are introduced, the unavailability drastically decreases and the overall system unavailability by summation is 45.4s.

6. Conclusions and future work

We considered Model-Based System Engineering methods managed across different disciplines and expertise applied to a real-size problem, COTS OBC for micro-satellite missions. The goal was to start from the same modelling base for (functional) applications and (architectural) platforms, then complete modelling with additional dedicated viewpoints for domain-specific properties. The common design environment chosen was Capella, augmented with Gemoc and Kitalpha Eclipse plug-ins. We focused on the study mixing analysis of interconnect congestion in local and distant communications, and on mission (un)availability due to solar radiations leading to failures and their mitigation.

Future work on (un)availability operational method with simulation shall reuse at least the operational semantic of execution components defined for congestion analysis. This will allow to precisely control the execution of safety mechanisms, to compute detection/mitigation delays and to manage concurrency in failure cascading. The introduced event dependencies using logical time constraints finally realized by physical time with adequate simulation steps will permit to precisely define exposition during satellite control missions and provide more accurate results. Consequently, stochastic simulation can also be performed, keeping in mind time scale diversity between failure rates, OBC mitigation delays and mission duration may lead to very long simulation time due to the required number of runs. We also need to document precisely the benefits of a new operational semantic implementation compared to the current Altarica capabilities since their concept for failure propagation is very close to our proposed approach.

7. Acknowledgement

The authors thank all people and partners involved in the ATIPPIC project managed by IRT Saint-Exupery in Sophia Antipolis, and in INRIA KAIROS team. The work performed in the ATIPPIC project context is supported by the French Research Agency (ANR) and by the industrial partners of IRT Saint-Exupéry Scientific Cooperation Foundation (FCS).

8. Références

[1] J. Whittle, J. Hutchinson and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE Software*, vol. 31, no. 3, pp. 79-85, 05 2014.

[2] P. Roques, "MBSE with the ARCADIA Method and the Capella Tool," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, 2016.

[3] L. Apvrille, W. Muhammad, R. Ameur-Boulifa, S. Coudert and R. Pacalet, "A UML-based Environment for System Design Space Exploration," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS 2006)*, Nice, France, Dec 2006.

[4] D. Genius and L. Apvrille, "Virtual Yet Precise Prototyping: An Automotive Case Study," in *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, Jan 2016.

[5] G. Carneiro, *NS-3 Network Simulator 3*, UTM Lab Meeting, 2010.

[6] A. Vargas and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, Marseille, France, 2008.

[7] M. Machin, L. Sagaspe et X. de Bossoreille, «SimfiaNeo, Complex Systems, yet Simple Safety,» chez *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, Toulouse, France, 2018.

[8] T. Prosvirnova, M. B. Batteux, P.-A. Brameret, A. CHERFI, T. Friedlhuber, J.-M. Roussel et A. Rauzy, «The AltaRica 3.0 Project for Model-Based Safety Assessment,» chez *4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013*, York, United Kingdom, 2013.

[9] A. Oueslati, P. Cuenot, J. Deantoni et C. Moreno, «System Based Interference Analysis in Capella,» *The Journal of Object Technology*, vol. 18, n° %12, p. 14:1, 2019.

[10] E. Bousse, T. Degueule, D. Vojtisek, T. Mayerhofer, J. Deantoni et B. Combemale, «Execution Framework of the GEMOC Studio (Tool Demo),» chez *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering*, Amsterdam, Netherlands, 2016.

[11] B. Combemale, J. Deantoni, M. E. Vara Larsen, F. Mallet, O. Barais, B. Baudry et R. France, «Reifying Concurrency for Executable Metamodeling,» chez *SLE - 6th International Conference on Software Language Engineering*, Indianapolis, IN, United States, 2013.

[12] J. Deantoni, P. Issa Diallo, C. Teodorov, J. Champeau et B. Combemale, «Towards a Meta-Language for the Concurrency Concern in DSLs,» chez *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Grenoble, France, 2015.

[13] European Cooperation for Space Standardization, ECSS-Q-HB-60-02A – Techniques for radiation effects mitigation in ASICs and FPGAs handbook, Noordwijk: ESA Requirements and Standards Division, 2016.

[14] International Organization for Standardization, *ISO/IEC 2382:2015 Information technology — Vocabulary*, 2015.